

YOUR NAME

SCHOOL

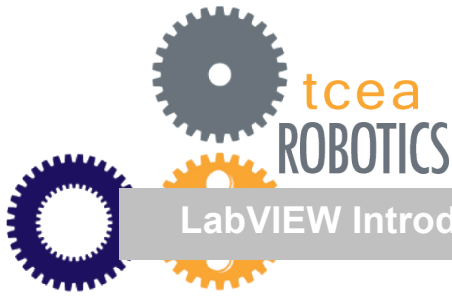
TCEA/TEXAS WORKFORCE COMMISSION ROBOTICS HIGH SCHOOL GRANT

PROGRAMMING RESOURCE



LEGO, THE LEGO LOGO, DUPLO, SOFT, AND MINDSTORMS EDUCATION LOGOS, THE BRICK AND KNOB CONFIGURATIONS AND THE MINIFIGURE ARE TRADEMARKS OF THE LEGO GROUP. © 2010 THE LEGO GROUP. ALL RIGHTS RESERVED. USE OF THIS SITE SIGNIFIES YOUR AGREEMENT TO THE TERMS OF USE.

LEGO EDUCATION NORTH AMERICA IS A JOINT VENTURE BETWEEN PITSCO, INC. AND THE EDUCATIONAL DIVISION OF THE LEGO GROUP.



LabVIEW™

LabVIEW is a programming language like C++ or Java but a little different than text-based languages because it is truly graphical. *LabVIEW* is composed of programming blocks that represent functions, inputs, and outputs that are connected with wires.

Industry Applications for *LabVIEW*

- Data acquisition and signal processing
- Controlling instruments
- Automation
- Measuring and controlling industrial systems
- Designing embedded systems
- Research and teaching

VI

VI is an acronym for Virtual Instrument, which is the term for a *LabVIEW* program.

To start programming, we can either select **Blank VI** or **Blank VI targeted to NXT** from the Getting Started screen. The difference between the two choices is that **Blank VI** will target program and run it on the computer only. **Blank VI targeted to NXT** will run only on a designated NXT.

When a VI is opened, we will see two windows. One is called **Block Diagram** and the other is called **Front Panel**.

The **Block Diagram** is where the graphical code is written, which consists of programming blocks that control motors, sensors, and so on that are connected to run sequentially by wires to create a flow of the program. The Block Diagram is where our programming code will go to tell our hardware what to do and when to do it.

The items we place on the Block Diagram are called **terminals** and **functions**.

The **Front Panel Diagram** is like the front panel of the NXT Brick. This is where we see buttons, displays, and indicators. The items we place on the Front Panel are called **controls** and **indicators**.

We can access the Block Diagram by selecting the Window drop-down menu and selecting **Show Block Diagram** or by using **Ctrl + E**. The shortcut **Ctrl + E** enables us to toggle back and forth from Front Panel and Block Diagram.

Terminal

The **Terminal** is an object that has data passing through.

Control

Control is an object that is located on the Front Panel and is used to enter data.

Indicator

Indicator is an object that is located on the Front Panel and displays data.

We start off in *LabVIEW* by selecting **Blank VI Targeted to NXT** because we want to create a program that runs on the NXT Brick and controls our MINDSTORMS® robot.

Next, we right-click the rectangle on the bottom left of our Front Panel or Block Diagram to **NXT Target: Unspecified** or **Find a specific NXT**.

The difference between the Main Application Instance and the Instance Targeting NXT is that different functions are available to us in NXT instance and therefore the palettes are slightly different. Some functions available in the Main Application Instance are not compatible with the NXT, so they do not appear on the palette in the NXT instance.

NXT I/O: Input /Output

This is the transfer of data to and from our NXT.

Open the NXT I/O sub-palette. Use the Context Help to assist in understanding each of the block icons.

For example, if you place a motor control icon on the Block Diagram and hover over the context help tool, you see two terminals on the block called **NXT input** and **NXT output**.

You will also find that there is a terminal for **Power Port** and one for **Output Port**. The **Output Port** is where we tell the program which motor to control. The **Power Port** tells our program the speed at which the motor should spin.

Polymorphic Selector: Drop-Down Menu

This drop-down menu is used to access a menu that can change the way a function acts.

Ctrl + E

Ctrl+E is used to toggle between the front panel diagram and block diagram.

Set a motor speed or designate motor port

To set a motor speed or designate a motor port, choose **Create** and then click **Constant**.

Automatic Tool Selector

Automatic Tool Selector is helpful turned on showing a bright green color will enable auto-wiring and enable you to manipulate the properties of an icon.

Wire Colors

- **Green Wire** - Boolean value providing yes/no – true/false values
- **Blue Wire** - Integer value (whole number)
- **Orange Wire** - Double-precision floating point value (numbers and decimals)
- **Pink Wire** - Connects inputs and outputs of icons to show a sequential flow of the program

Random Number

A **random number** does not follow a plan, order, or purpose. By default, it creates a random number between 0 and 100.

While Loop

A **While Loop** is the part of the program that is located inside the loop until a specific condition or behavior is met.

A While Loop contains two terminals. The one on the left is called the **iteration terminal**. The iteration terminal is the blue box with “i” in it. The one on the right is a green box called the **conditional terminal**.

Boolean value

We wire the conditional terminal to tell it when the program should exit. A **Boolean** value is a true or false value, and depending on the setup of the while loop, it will either continue the loop when the value is true or stop the loop when the value is false.

The **iteration terminal** outputs the number of times a loop will run, but the very first time it runs, the output is 0. The first time through the loop, the value of “i” is 0. The second time through the loop, the value is 1, and so on. For example, if you wanted a program to loop four times, the “i” would be equal to 3.

If we want to create a loop that runs forever and we are in the stop sign mode, we want to constantly input a false value into the terminal. Remember, if true means stop and false means go, if it is always false then it will go forever. To create the loop that will run forever, we need to always have a false there.

Wait For

A **Wait for** is a function that causes a timed delay in the program before letting the program begin.

Parallelism

Parallelism is when many functions are carried out simultaneously.

Deterministic

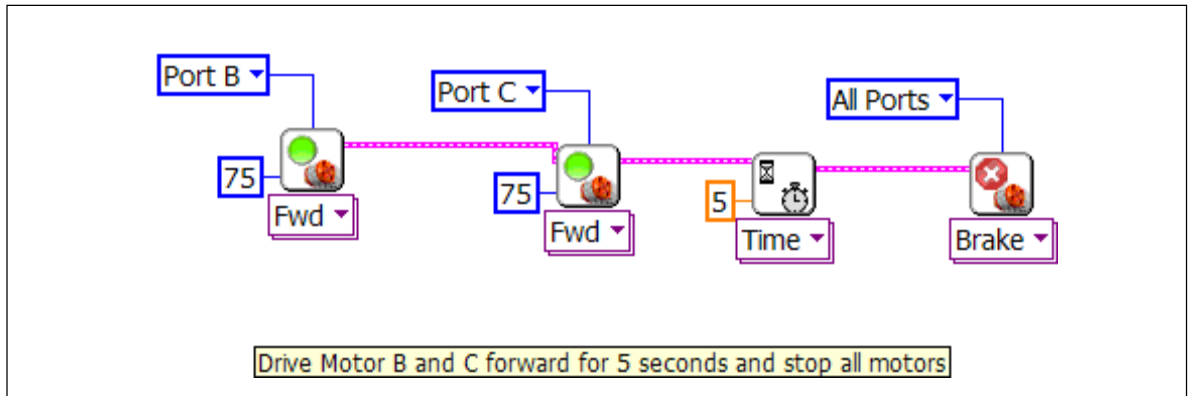
Deterministic is how all things occur in the order as intended by the user.

Case Structures

The case structure in **LabVIEW** is very similar to the if/else statement found in many other programming languages. A case structure enables us to do one thing in one situation and something different in another situation. A good example of this is if our robot is near a wall. If the robot is near a wall, we want it to stop. If the robot is not near a wall, we want it to keep going.

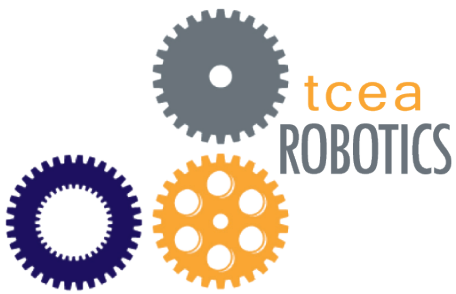
By default, a case structure is expected to take in a **Boolean** input. Therefore, we get two cases: a true case and a false case.

Program # 1

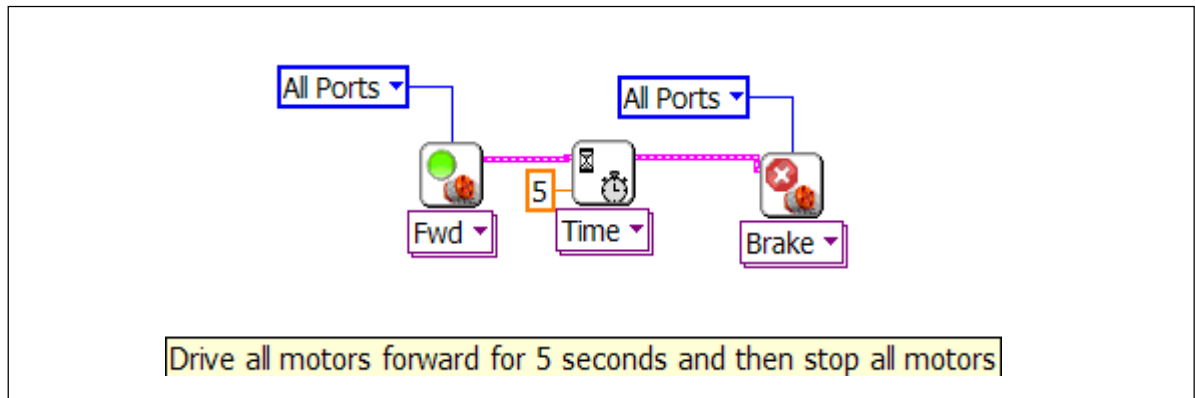


- Select 2 Motor Blocks (found in Function Palette - NXT I/O) - forward
- Assign Port B (found in NXT I/O - Complete - Motor Block) to first motor block and Port C to second motor block
- Assign a power of 75 (found in NXT Programming - Numeric - Numeric Constant) to each motor blocks
- Select Wait Block (NXT I/O) - time
- Assign a period of 5 seconds (NXT Programming - Numeric - Numeric Constant)
- Select Motor Block (NXT I/O)
- Configure in drop down menu Motor Off - Brake
- Assign All Ports (NXT I/O - Complete - Motor Block)
- Wire
- Deploy program

NOTES:

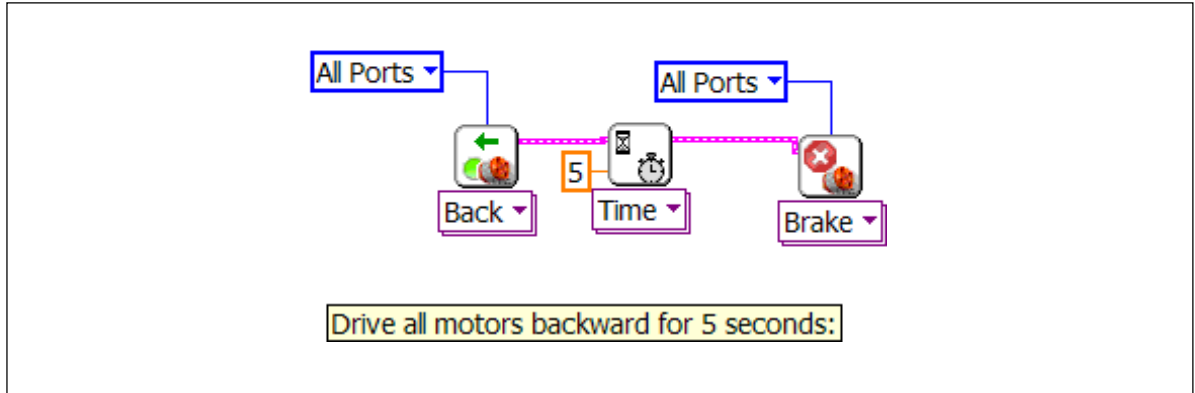


Program # 2



- Select a Motor Block (Function Palette - NXT I/O) - forward
- Assign All Ports (NXT I/O - Complete - Motor Block - select all Ports)
- Select Wait Block (NXT I/O) - time
- Assign a period of 5 seconds (NXT Programming - Numeric - Numeric Constant)
- Select Motor Block (NXT I/O)
- Configure in drop down menu Motor Off - Brake
- Assign All Ports (NXT I/O – Complete) to motor block
- Wire
- Deploy program

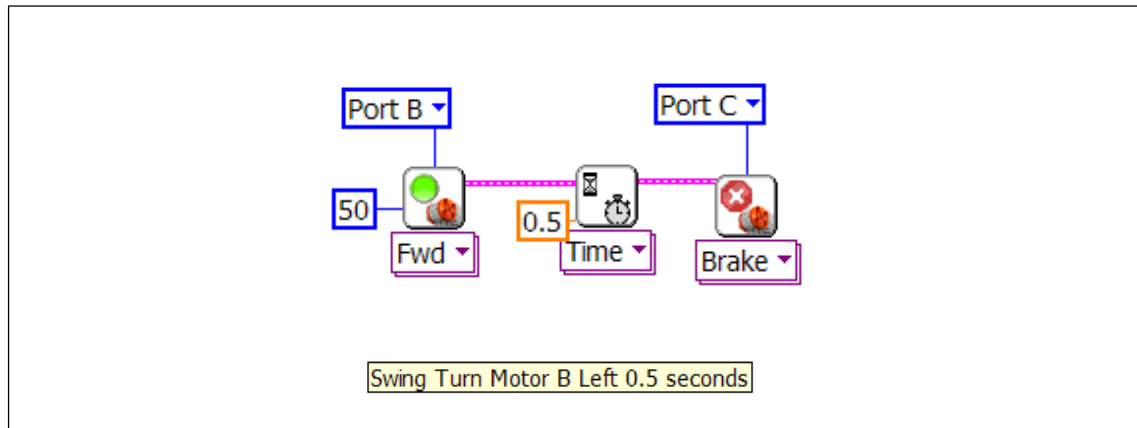
NOTES:



- Select a Motor Reverse Block (Function Palette - NXT I/O - Complete - Motors)
- Assign All Ports (NXT I/O - Complete) to motor block
- Select Wait Block (NXT I/O) - time
- Assign a period of 5 seconds (NXT Programming - Numeric - Numeric Constant)
- Select Motor Block (NXT I/O)
- Configure in drop down menu Motor Off - Brake
- Assign All Ports (NXT I/O - Complete) to motor block
- Wire - Check program - Deploy program

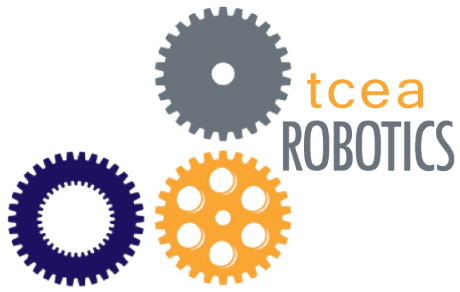
NOTES:

Program # 4

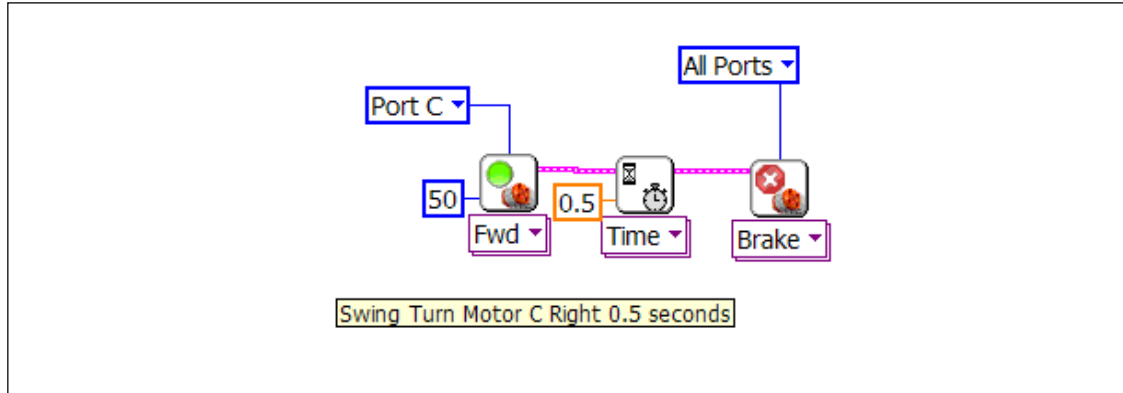


- Select a Motor Block (Function Palette - NXT I/O) - forward
- Assign Port B (NXT I/O - Complete) to motor block
- Assign a power of 50 (found in NXT Programming - Numeric - Numeric Constant) to motor block B
- Select Wait Block (NXT I/O) - time
- Assign a period of 5 seconds (NXT Programming - Numeric - Numeric Constant)
- Select Motor Block (NXT I/O)
- Assign Port C (NXT I/O - Complete) to motor block
- Configure in drop down menu Motor Off - Brake
- Wire - Check program - Deploy program

NOTES:



Program # 5

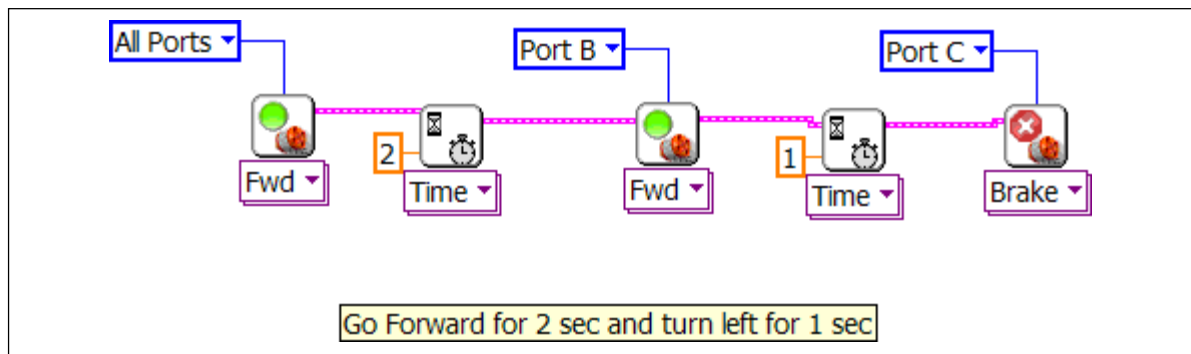


Select a Motor Block (Function Palette - NXT I/O) - forward

- Assign Port C (NXT I/O - Complete) to motor block
- Assign a power of 50 (NXT Programming - Numeric - Numeric Constant) to motor block C
- Select Wait Block (NXT I/O) - time
- Assign a period of 5 seconds (NXT Programming - Numeric - Numeric Constant)
- Select Motor Block (NXT I/O)
- Assign Port B (NXT I/O - Complete) to motor block
- Configure in drop down menu Motor Off - Brake
- Wire - Check program - Deploy program

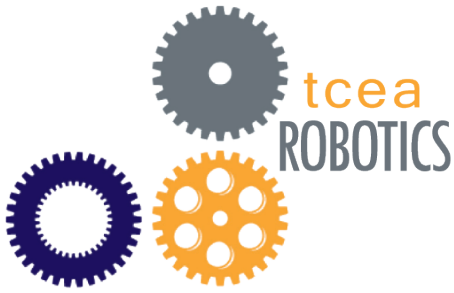
NOTES:

Program # 6

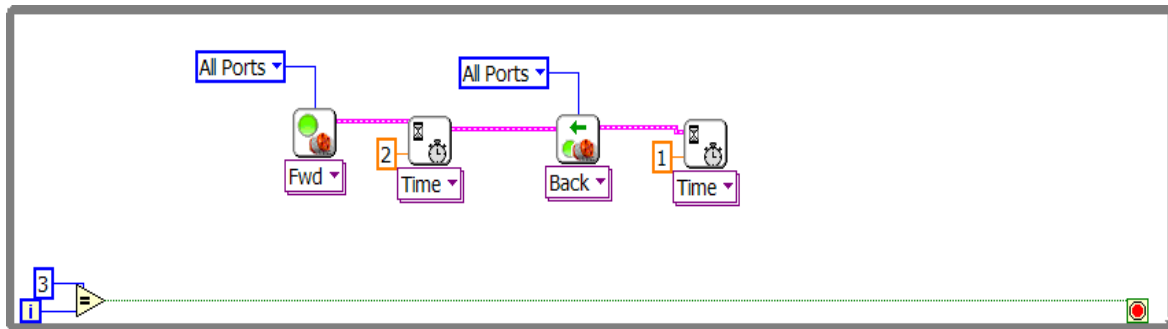


- Select a Motor Block (Function Palette - NXT I/O) - forward
- Assign All Ports (NXT I/O - Complete) to motor block
- Select Wait Block (NXT I/O) - time
- Assign a period of 2 seconds (NXT Programming - Numeric - Numeric Constant)
- Select a Motor Block (Function Palette - NXT I/O) - forward
- Assign Port B (NXT I/O - Complete) to motor block
- Select Wait Block (NXT I/O) - time
- Assign a period of 1 seconds (NXT Programming - Numeric - Numeric Constant)
- Select Motor Block (NXT I/O)
- Assign Port C (NXT I/O - Complete) to motor block
- Configure in drop down menu Motor Off -Brake

NOTES:



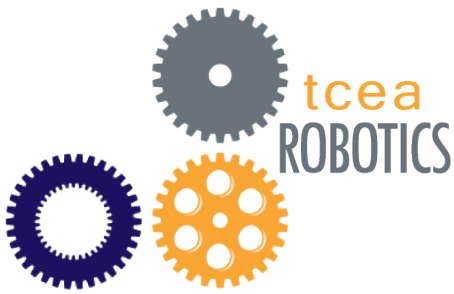
Program # 7



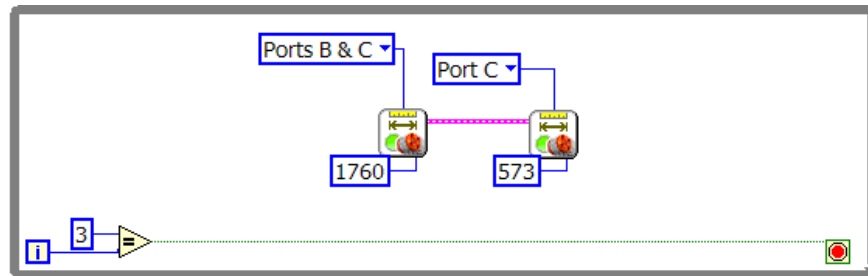
Program inside while loop is repeated 4 times. NXT moves forward for 2 seconds and backward for 1 second four times.

- Select a Motor Block (NXT I/O - forward)
- Assign All Ports (NXT I/O - Complete) to motor block
- Select Wait Block (NXT I/O) - time
- Assign a period of 2 seconds (NXT Programming - Numeric - Numeric Constant)
- Select a Motor Reverse Block (NXT I/O - Complete - Motors)
- Assign All Ports (NXT I/O - Complete) to motor block
- Select Wait Block (NXT I/O) - time
- Assign a period of 1 second (NXT Programming - Numeric - Numeric Constant)
- Draw a While Loop (NXT Programming - Structures) around all blocks
- Select an Equal Block (NXT Programming - Comparison) and place in bottom left of loop, near Loop Iteration
- Assign a numerical value of 3 (NXT Programming - Numeric - Numeric Constant) to the Equal
- Wire Loop Iteration to Y terminal (bottom left) on Equal Block and numeral 3 to X terminal (top left) on Equal Block
- Wire Equal Block $x = y?$ terminal (right) to Loop Condition (bottom right of Loop)

NOTES:



Program # 8

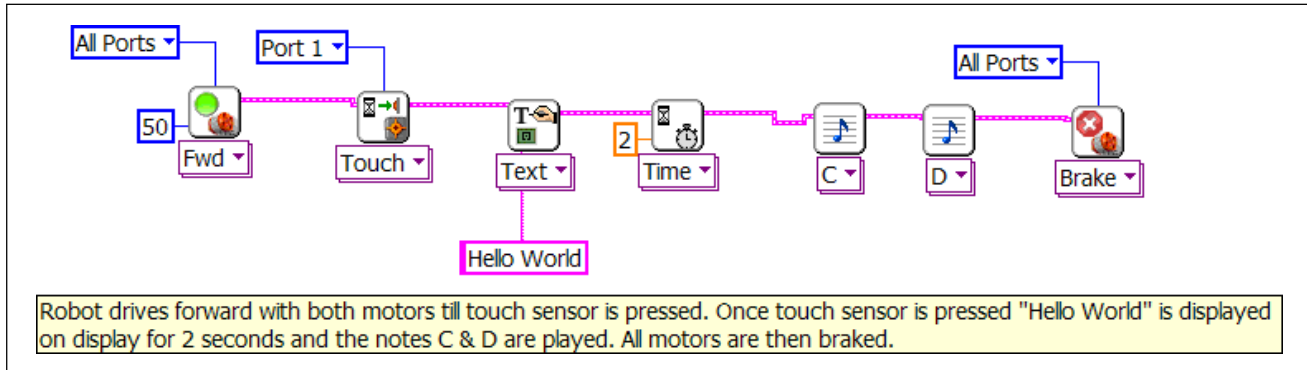


Robot travels in a perfect square. Motors B & C both travel forward for 1760 degrees and then turn right with motor C traveling for 573 degrees

- Select a Drive Distance Block (NXT I/O - Complete - Motors)
- Assign Ports B&C (NXT I/O - Complete) to motor block
- Assign a degree of 1760 (NXT Programming - Numeric - Numeric Constant)
- Select a Drive Distance Block (NXT I/O - Complete - Motors)
- Assign Port C (NXT I/O - Complete) to motor block
- Assign a degree of 573 (NXT Programming - Numeric - Numeric Constant)
- Draw a While Loop (NXT Programming - Structures) around all blocks
- Select an Equal Block (NXT Programming - Comparison) and place in bottom left of loop, near Loop Iteration
- Assign a numerical value of 3 (NXT Programming - Numeric - Numeric Constant) to the Equal Block
- Wire Loop Iteration to Y terminal (bottom left) on Equal Block and numeral 3 to X terminal (top left) on Equal Block
- Wire Equal Block $x = y?$ terminal (right) to Loop Condition (bottom right of Loop)

NOTES:

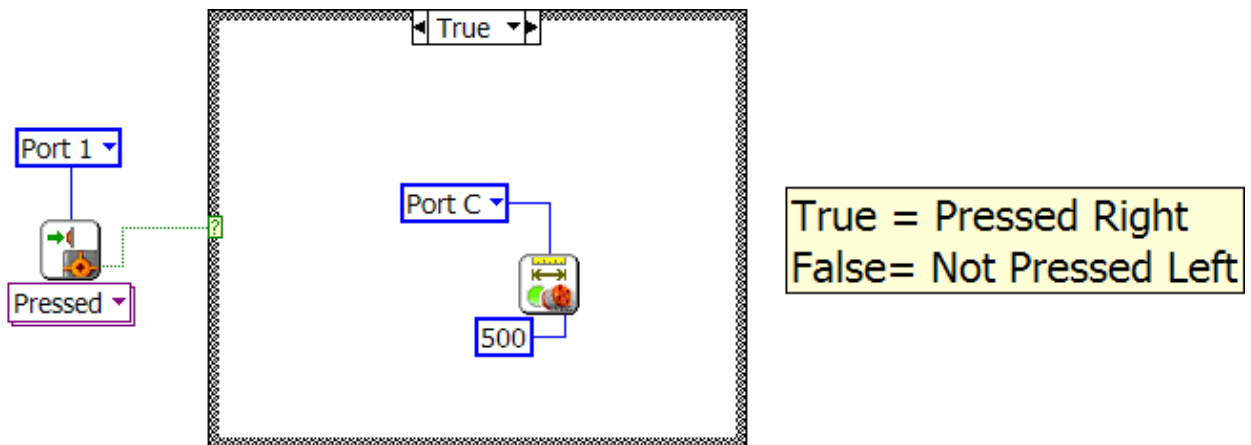
Program # 9



- Select a Motor Block (NXT I/O) - forward
- Assign All Ports (NXT I/O - Complete) to motor block
- Assign a power of 50 (Right Click on Motor power input (bottom left) - Create - Constant)
- Select a Wait Block (NXT I/O) - Wait for Touch - pressed
- Assign Port 1 (Right Click on port (top) - Create - Constant)
- Select a Display Block (NXT I/O) - text
- Type Hello World (Right Click on Text port(bottom left) - Create - Constant)
- Select Wait Block (NXT I/O) - time
- Assign a period of 2 seconds (Right Click on Time port (bottom left) - Create - Constant)
- Select Sound Block (NXT I/O) - tone
- Assign a C Tone (Left Click on Tone drop down menu - Note)
- Select Sound Block (NXT I/O) - tone
- Assign a D Tone (Left Click on Tone drop down menu - Note)
- Select Motor Block (NXT I/O)
- Configure in drop down menu Motor Off - Brake
- Assign All Ports (NXT I/O - Complete) to motor block

NOTES:

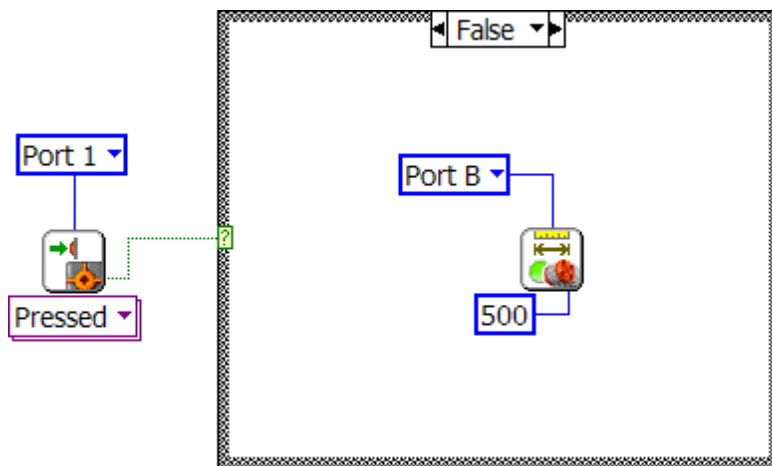
Program # 10



Introduces Cases

When touch sensor is pressed, as a true case then the robot turns right as motor c travels 500 degrees

When touch not sensor is pressed, as a false case then the robot turns left as motor B travels 500 degrees



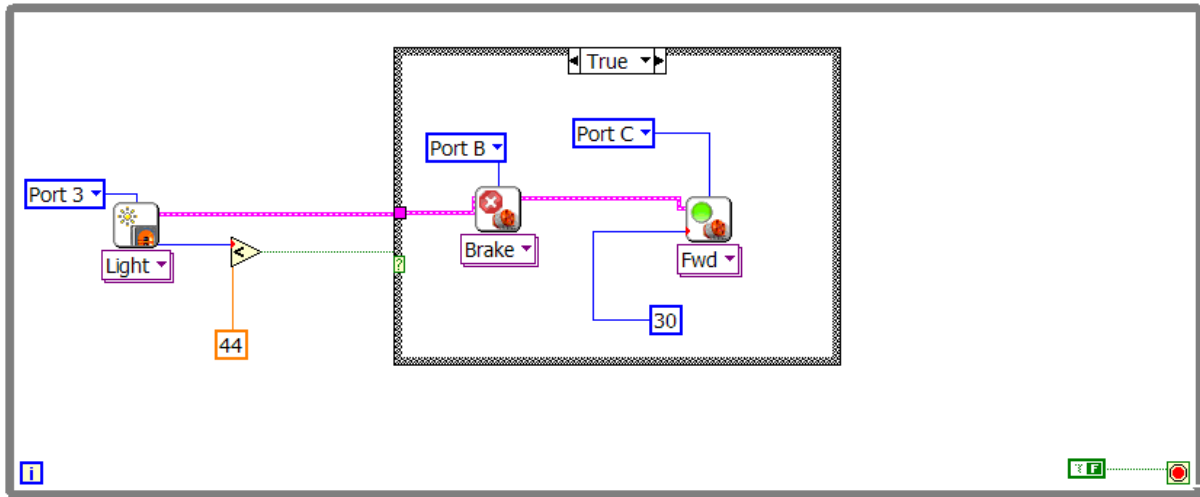
- Select Sensor Block (NXT I/O) - touch sensor
- Assign Port 1 (Right Click on Port (top) - Create - Constant)
- Draw a Case Structure (NXT Programming - Structure)
- Wire Case Selector to Touch Sensor Yes/No (bottom right)
- Inside of True Case Structure insert a Drive Distance Block (NXT I/O - Complete - Motors)

- Assign Port C (Right Click on Output Port (top) - Create - Constant)
- Assign a distance of 500 degrees (Right Click on Distance Input (bottom) - Create - Constant)
- **Change the Case Structure Selector Label to False using the drop down menu**
- Inside of the False Case Structure insert a Drive Distance Block (NXT I/O - Complete - Motors)
- Assign Port B (Right Click on Output Port (top) - Create - Constant)
- Assign a distance of 500 degrees (Right Click on Distance Input (bottom) - Create - Constant)

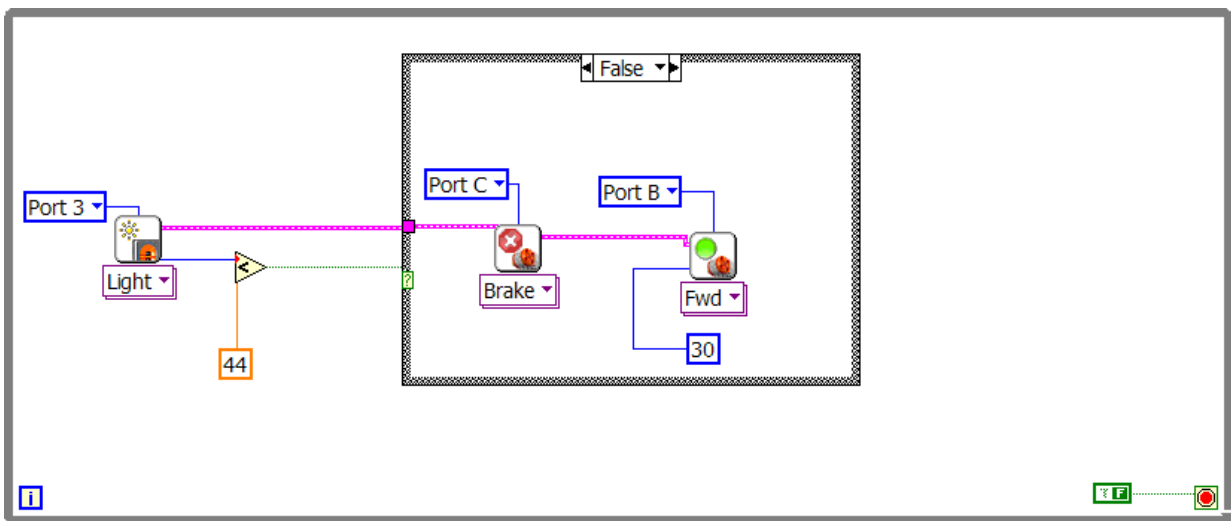
NOTES:

Program # 11

True = B off - C on turns Right
False = C off - B on turns Left

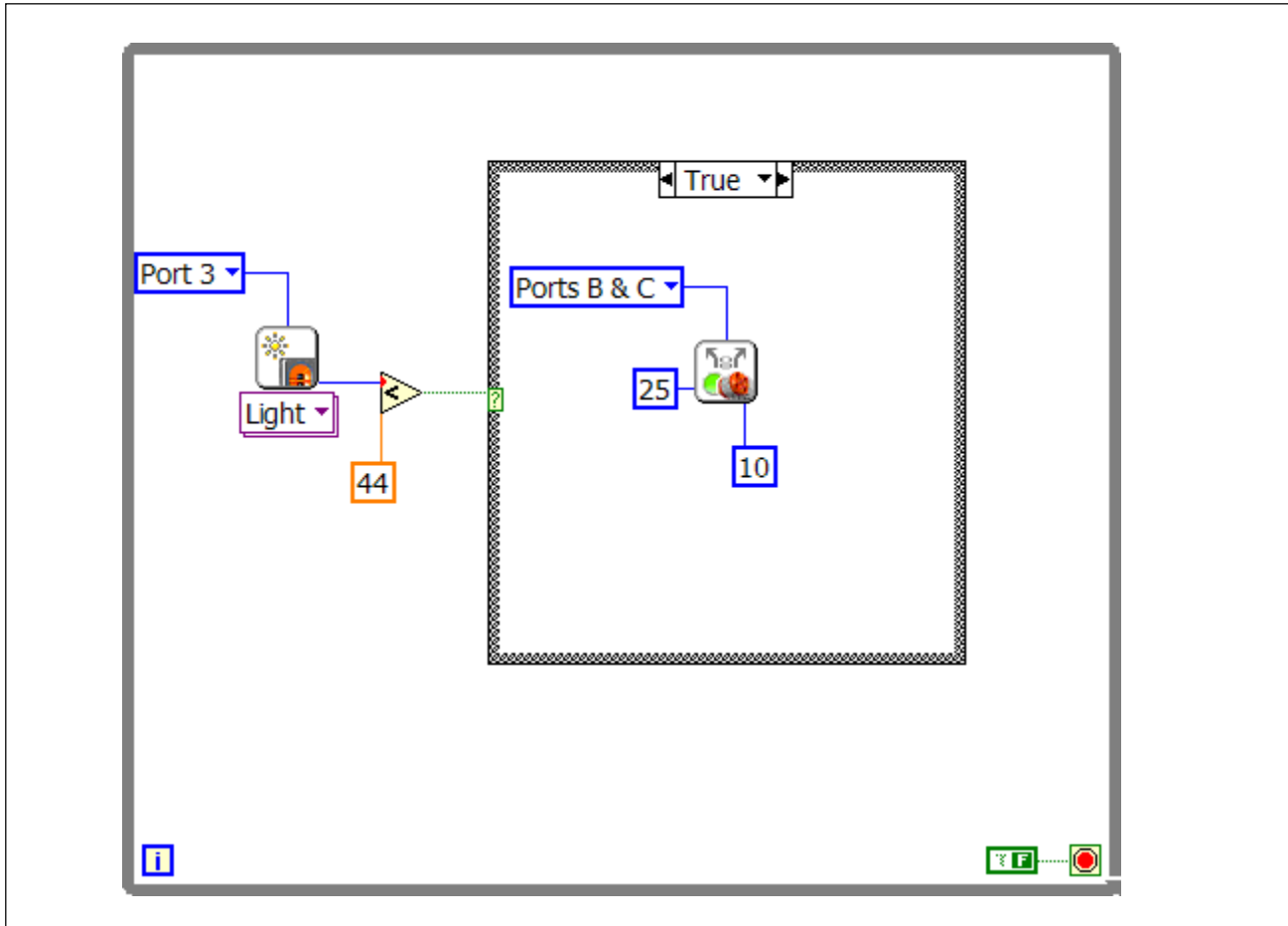


When light sensor reads a light level less than 44 (true case) = Motor B is turned off and Motor C moves forward at a speed of 30
When light sensor reads a light level above 44 (false case) = Motor B moves forward at a speed of 30 and Motor C is turned off



- Select Sensor (NXT I/O) light sensor (Left Click on sensor drop menu - read light - LED On)
- Assign Port 3 (Right Click on Port (top) - Create - Constant)
- Add Less Block to the right of Light Sensor Block (NXT Programming - Comparison)
- Assign a threshold value of 44 (Right Click on Y Terminal (bottom left) - Create - Constant)
- Connect Light Sensor Yes/No Output (bottom right) to Less Block X Terminal (top left)
- Draw Case Structure (NXT Programming - Structures) - true (the default)
- Inside of the True Case Structure insert a Motor Block (NXT I/O)
- Configure in drop down menu Motor Off – Brake
- Assign Port B (Right Click on Output Port (top) - Create - Constant)
- Inside of the True Case Structure insert a Motor Block (NXT I/O) - forward
- Assign Port C (Right Click on Output Port (top) - Create - Constant)
- Assign a power of 30 (Right Click on power input (bottom left) - Create - Constant)
- Connect Light Sensor Block (right side input port) to True Case Structure (left side)
- Continue the wire inside the True Case Structure and connect the two motor blocks
- Connect Less Block X<Y? terminal (right side) to True Case Structure Case Selector
- **Change True Case Structure to False (Selector Label drop menu)**
- Inside of the False Case Structure insert a Motor Block (NXT I/O)
- Configure in drop down menu Motor Off - Brake
- Assign Port C (Right Click on Output Port (top) - Create - Constant)
- Inside of the False Case Structure insert a Motor Block (NXT I/O) - forward
- Assign Port B (Right Click on Output Port (top) - Create - Constant)
- Assign a power of 30 (Right Click on power input (bottom left) - Create - Constant)
- Continue the wire inside the False Case Structure and connect the two motor blocks
- Draw a While Loop (NXT Programming - Structures) around Case Structure and Light Sensor Block
- Select a False Constant (NXT Programming - Boolean) place near Loop Condition in bottom right
- Connect False Constant Block to Loop Condition

Program # 11B (NXT – Easy Line following - moving toward the line)

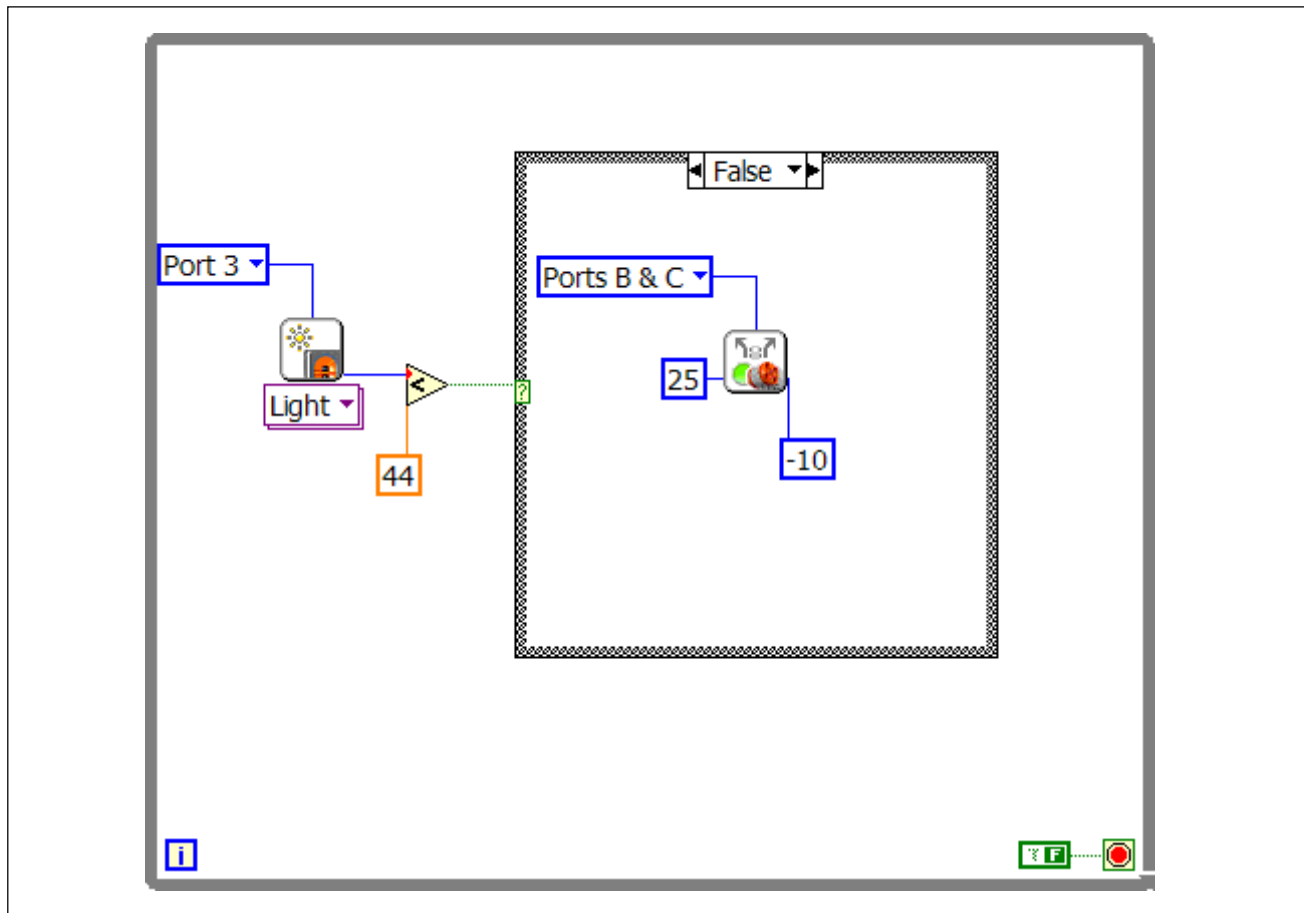


To start programming, choose a Blank VI targeted to NXT

1. The idea behind line following is to always move towards the line, specifically the edge of the line. When the light sensor reads dark, it is on the line so the robot should move forward and left until the sensor reads light. Then the robot should move forward and right until the sensor readings dark and is back onto the line.
2. To create the True Case above, start by drawing a Case Structure from NXT Programming Palette> Structures sub-palette to the workspace.
3. The Case Structure should switch based on the light sensor, so drag a Read Sensor from the NXT I/O palette and change it to Read Light LED On. Add a Less Than from NXT Programming > Comparison with a constant of 44, then wire them to the Case Structure. You may need to adjust your threshold to better suit your conditions.

4. Add a Port Constant to the top of the light sensor and set it to the port that the light sensor is plugged in to. The True Case in the structure will be run when the light sensor receives a reading less than 44 (dark) and the False Case will run when the light sensor receives a reading greater than or equal to 44 (light).

5. Drag a Steering On from the NXT I/O>Complete>Motors sub-palette into the True Case Structure. Create a Constant for Power on the left, Steering on the bottom, and Ports on the top. Set the Ports constant to the two ports your motors are connected to. Set the power to 25 and the steering to 10.



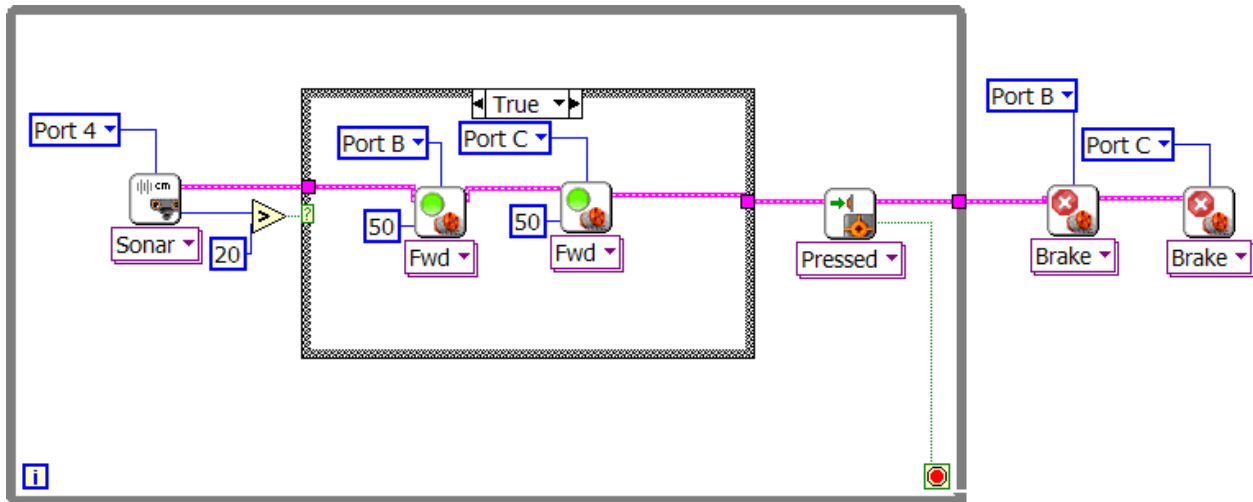
6. To create the False Case (Above) drag your mouse to select all the items in the True Case statement, then press CTRL- C to copy them. Change the Case Structure to False and press CTRL – V to paste the contents from the True Case Structure.

7. Change the Steering Constant in the False Case to -10. The robot should move forward and turn left when the light sensor detects dark, and move forward and turn right when the light sensor detects light.

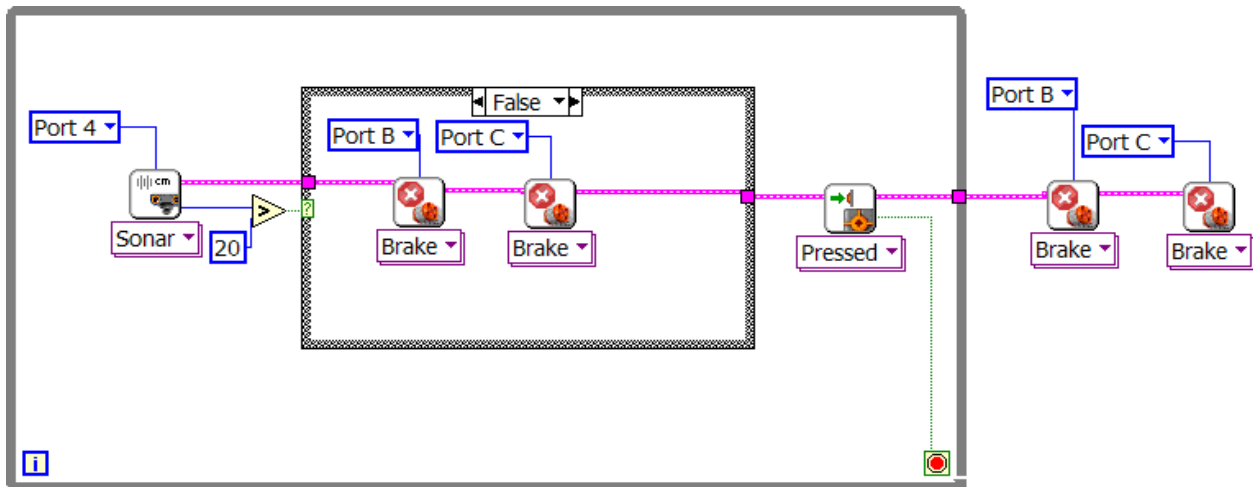
8. Repeating this behavior will make the robot follow the line. Draw a While Loop from the NXT Programming>Structures sub-palette around everything in the environment. Add a Boolean Constant (False) to the stop condition. Now the VI repeatedly checks the light sensor and moves toward the line.

9. Deploy the VI to the NXT, unplug the robot and run the program.

Program # 12

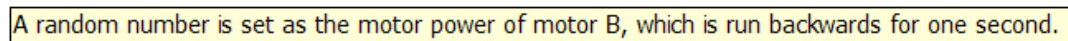


The NXT robot moves forward until the Ultrasonic Sensor senses something in front of the robot, it then stops. Should the object be removed from the robot's path then the robot continues to move. Pressing the touch sensor stops the program.



- Select Sensor Block (NXT I/O) ultrasonic sensor (Left Click on sensor drop menu – read ultrasonic)
- Assign Port 4 (Right Click on Port (top) – Create – Constant)
- Add Greater Block to the right of Light Sensor Block (NXT Programming – Comparison)
- Assign a threshold value of 20 (Right Click on Y Terminal (bottom left) – Create – Constant)

- Connect Ultrasonic Sensor Yes/No Output (bottom right) to Greater Block X Terminal (top left)
- Draw Case Structure (NXT Programming – Structures) true (the default)
 - Inside of the True Case Structure insert a Motor Block (NXT I/O) forward
 - Assign Port B (Right Click on Output Port (top) – Create – Constant)
 - Assign a power of 50 (Right Click on power input (bottom left) – Create – Constant)
 - Inside of the True Case Structure insert a Motor Block (NXT I/O) forward
 - Assign Port C (Right Click on Output Port (top) – Create – Constant)
 - Assign a power of 50 (Right Click on power input (bottom left) – Create – Constant)
 - Connect Ultrasonic Sensor (right side input port) to True Case Structure (left side)
 - Continue the wire inside the True Case Structure and connect the two motor blocks
- Connect Greater Block X<Y? terminal (right side) to True Case Structure Case Selector
- **Change True Case Structure to False (Selector Label drop menu)**
 - Inside of the False Case Structure insert a Motor Block (NXT I/O)
 - Configure in drop down menu Motor Off – Brake
 - Assign Port B (Right Click on Output Port (top) – Create – Constant)
 - Inside of the False Case Structure insert a Motor Block (NXT I/O)
 - Configure in drop down menu Motor Off – Brake
 - Assign Port C (Right Click on Output Port (top) – Create – Constant)
 - Continue the wire inside the False Case Structure and connect the two motor blocks
 - Select Sensor Block (NXT I/O) touch sensor and place to the right of Case Structure
 - Assign Port 1 (Right Click on Port (top) – Create – Constant)
 - Continue wire from motor blocks to right side of Case Structure and to the Touch Sensor
 - Draw a While Loop (NXT Programming – Structures) around Case Structure, Ultrasonic Sensor Block, and Touch Sensor Block
 - Connect Touch Sensor (bottom right) to Loop Condition
 - Continue wire from Touch Sensor to the right side of While Loop
 - Outside of the Loop place a Motor Block (NXT I/O)
 - Configure in drop down menu Motor Off – Brake
 - Assign Port B (Right Click on Output Port (top) – Create – Constant)
 - Outside of the Loop place a Motor Block (NXT I/O)
 - Configure in drop down menu Motor Off – Brake
 - Assign Port C (Right Click on Output Port (top) – Create – Constant)
 - Continue wire from While Loop to both motor brakes



Program # 14

Counting Program X + Y or Subtracting Program X-Y

To help us understand how Case Structures work, create a program in which one Case adds two numbers and in the other Case subtracts the two numbers.

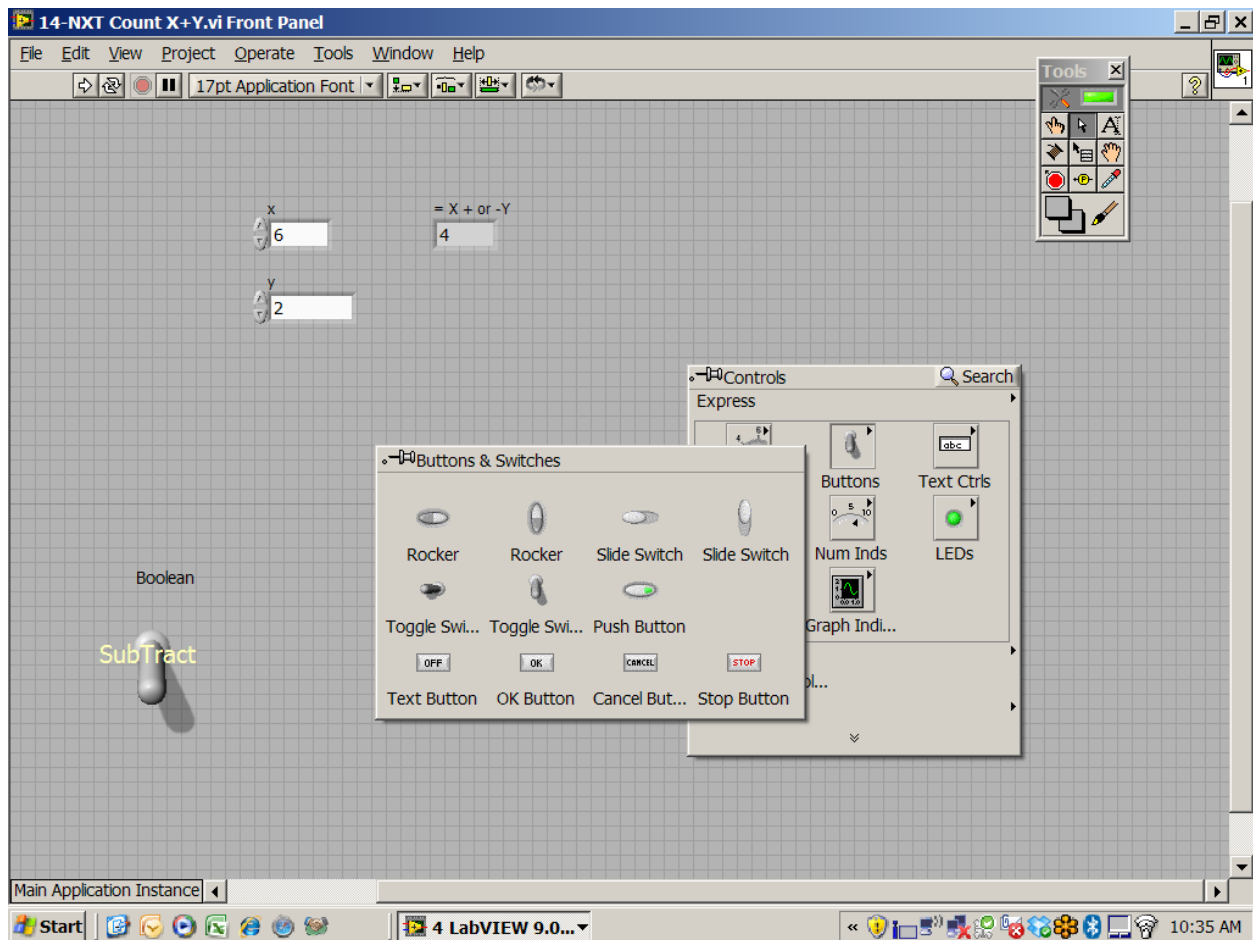


Figure 1.

1. Start this program with a New VI targeted to NXT.
2. Begin by placing a Case Structure onto the Block Diagram. We will add the two numbers in the True Case and subtract the two numbers in the False Case.
3. To indicate whether we are in an add mode or in a subtract mode go to the front panel and create a toggle switch. To place a toggle switch, right click on the front panel, select Boolean pallet, and click Vertical Toggle Switch.

4. We can resize the toggle switch so it is easier to use. (*Pull on one of the corner dots to enlarge*)

5. Let's right click the toggle switch, select Visible Items, and click Boolean Text. This creates a label to tell us if the toggle switch is outputting a True or False Value. When it is in the down position, the label says "OFF", meaning that the switch is outputting a False Value. We can click the switch when the cursor turns into a pointer, and the switch will go to the up position with the label saying "ON". This is the position in which the switch **outputs** a True Value.

6. We can now double-click the "On" to rename it as "Add" because we will be adding in the True Case. I would also recommend increasing the font size as well as changing to yellow so it can be easily seen. Likewise, we can rename the "Off " to "Subtract" because we will be subtracting in the False case. (Use the edit text tool to rename text.)

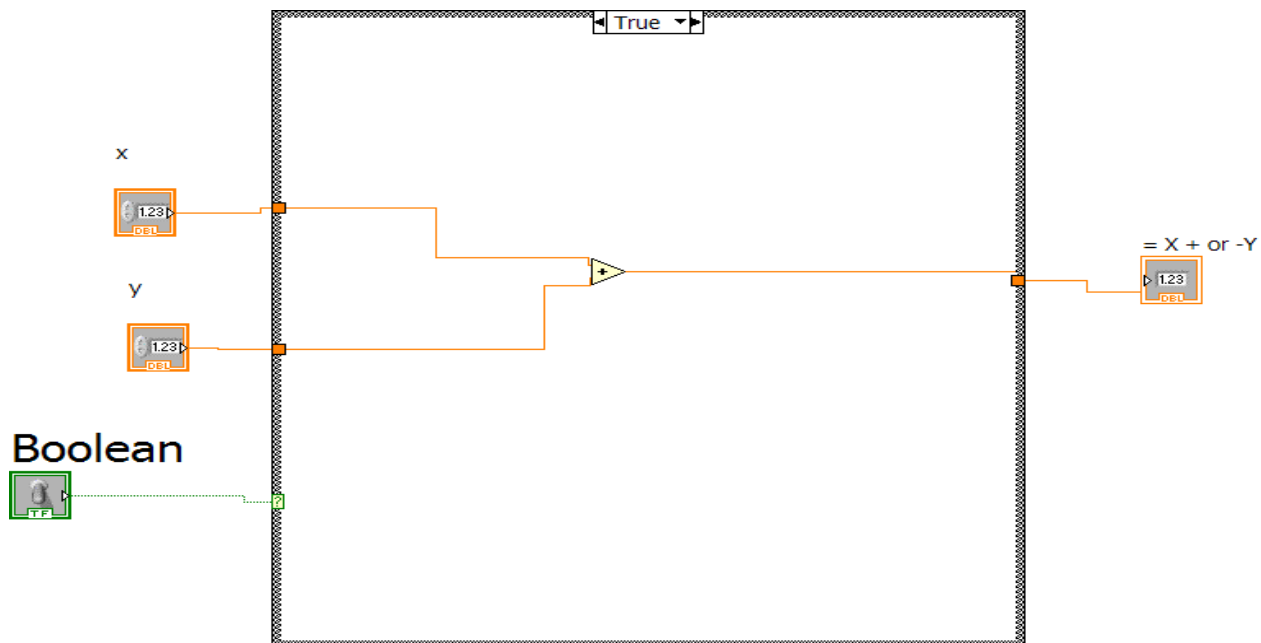


Figure 2

7. If we switch over to the block diagram, we will see the toggle switch has appeared. We will wire this to the case selector of the case structure.

8. Since we want to add in the true case, we will place an Add function inside the case structure. We are subtracting inside the false case, so we will place a Subtract function inside the False Case. The Subtract and the Add functions are located under the **NXT Programming**, Numeric sub pallet.

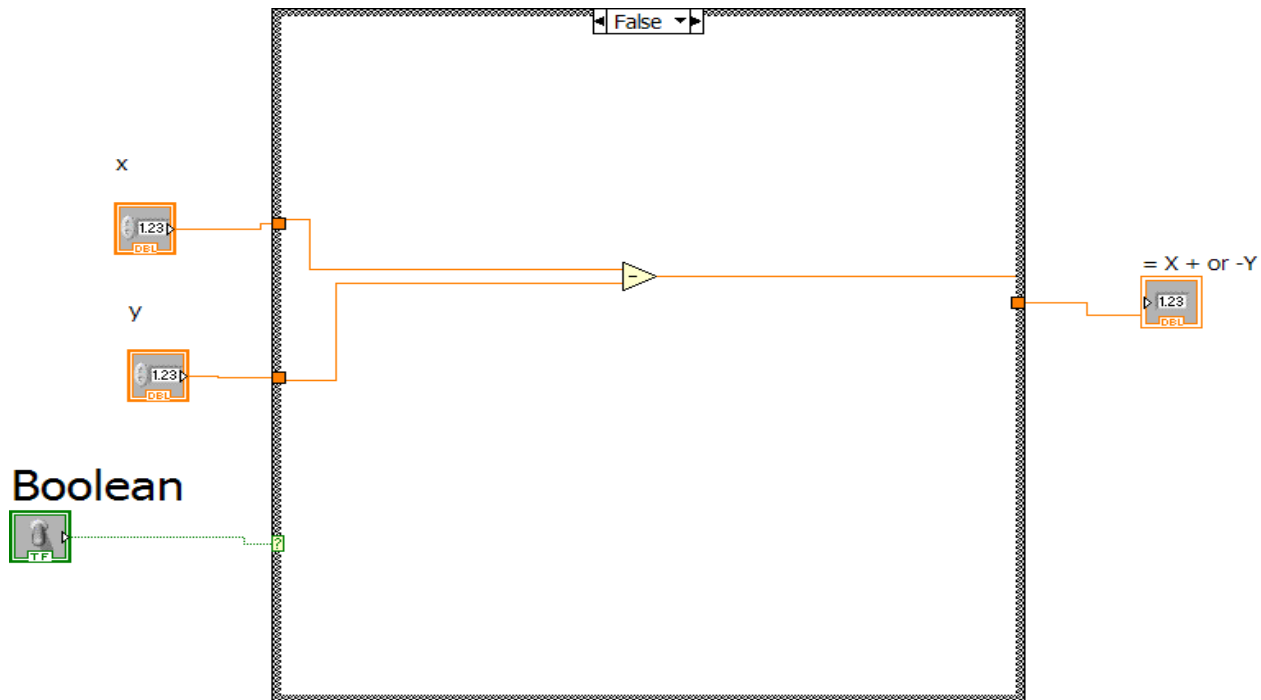


Figure 3

9. Go back to the Front Panel and create two Numeric Controls that will allow the two values that will be added/subtracted to be entered. A Numeric Indicator will also be needed to display the result. These Numeric Controls and Indicators can be accessed from Controls > Numeric sub-pallet. See Figure 1.

10 . Right click on each Numeric label, select properties and change visible numeric to “X”, numeric 2 to “Y”, and numeric 3 to “= X + Y” or “X-Y”.

11. Go back to the block diagram and place the two Controls and the Indicator in the indicated location as shown in Figure 2 and 3.

12. Wire the two controls to the inputs of the Add Function, and wire the output to the Add function to the Indicator.

13. In the False Case wire the two controls to the inputs of the Subtract function, and wire the output of the Subtract function to the Indicator.

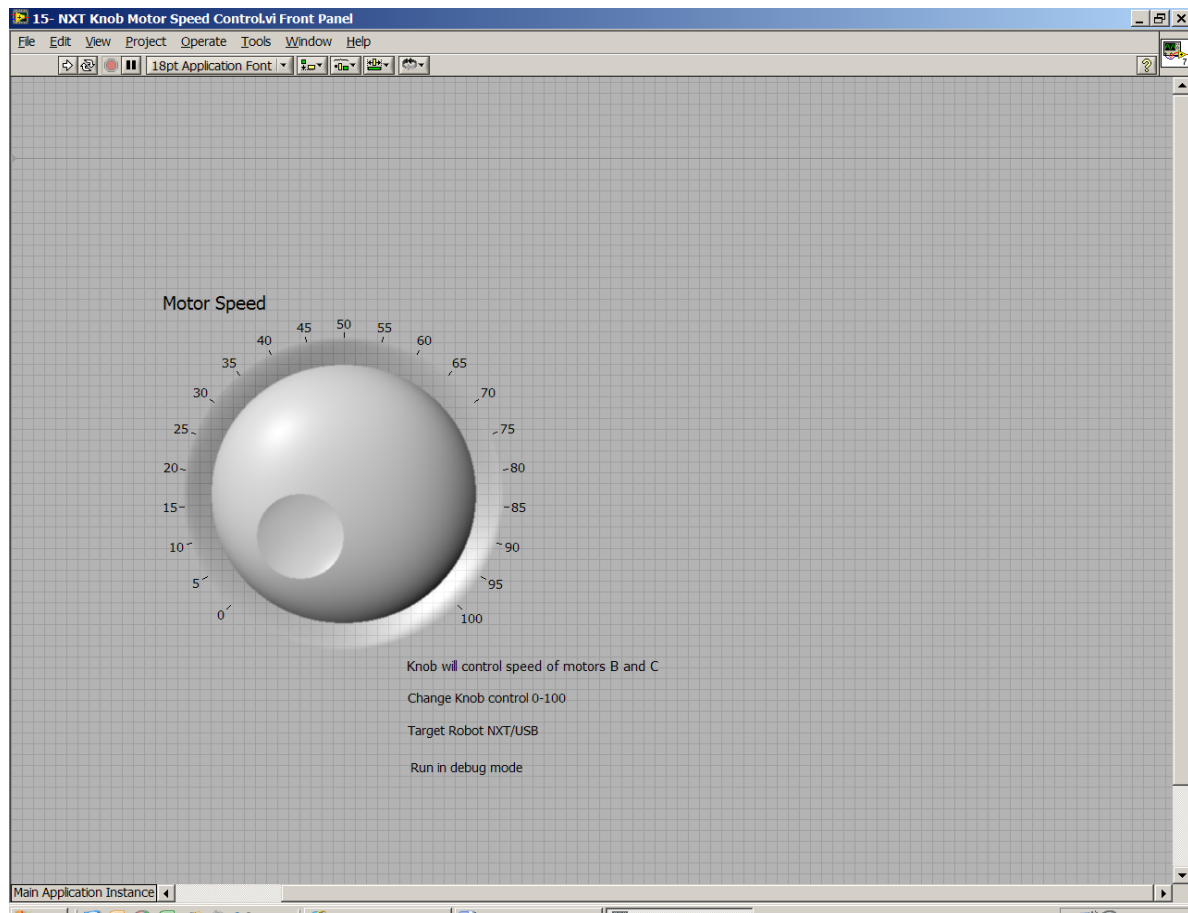
14. To run the program in the main application select Target to Computer from the File menu.

15. Go back to Front Panel (figure 1) and enter two values in our controls. Let’s try 7 and 2 and put the switch in add mode. Click the Run button and our Indicator will display 9.

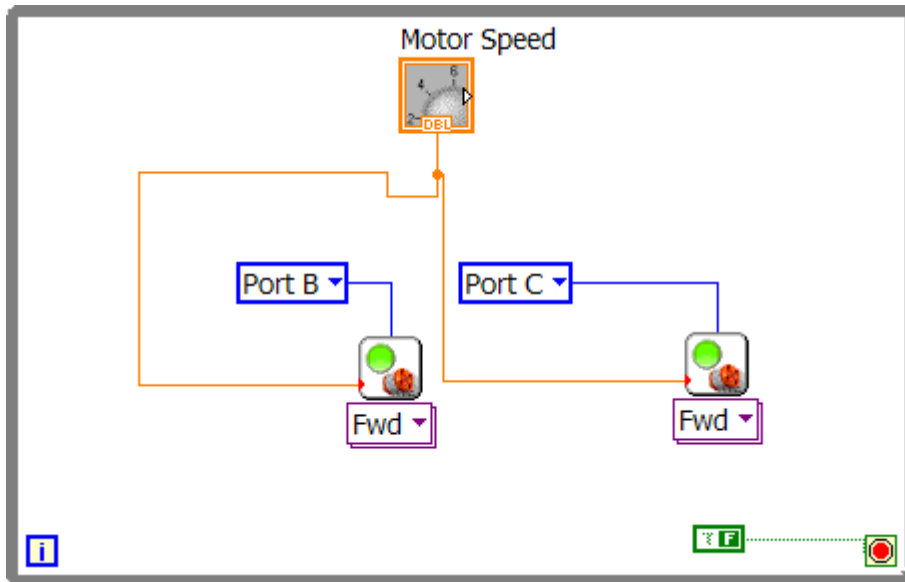
16. Now let’s use this program to subtract. Click the toggle switch to switch to the subtract mode. We can run this program and see it calculates 7-2, and displays the result.

Program # 15 NXT Knob Motor Control

1. Begin with a blank VI targeted to an NXT. Have your robot connected via USB and turned on.
2. While in Front Panel, right click to find numeric controls sub-palette and then select and place a knob on the front palette.
3. Enlarge the size of the knob by pulling on one of the corner dots on the knob.
4. Right click on the largest number of the knob and select properties.
5. In the Properties Window Appearance Tab change the label name from Knob to Motor Speed. Ensure the visible box is checked.
6. In the Scale Tab change the scale range maximum from 10 to 100 and click OK to save properties changes.

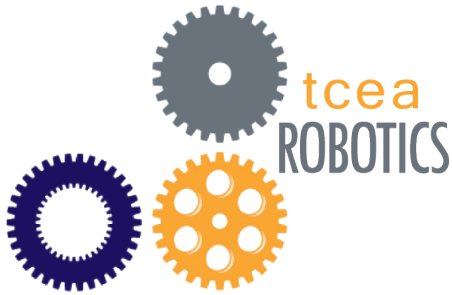


7. Switch to the Block Diagram. Move Motor Speed block toward the top middle of the diagram as shown below.



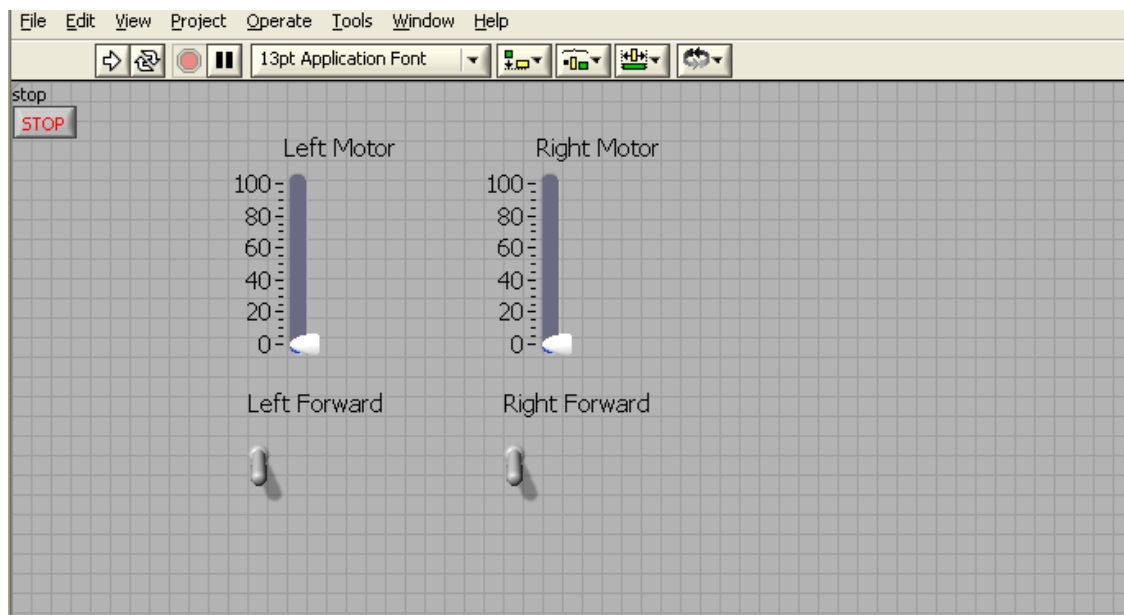
8. To the right and left of the Motor Speed block add a Motor Control by right clicking on Block Diagram and choose from NXT I/O.
9. Create Constant on the output ports of the Motor Control block on left Port B and then Create Constant on the output ports of the Motor Control block on right Port C.
10. Wire both B and C motors starting from the power level terminals to the Motor speed knob.
11. So that the robot can collect speed level more than once draw a While Loop from the Structures palette found in NXT Programming around the Motor Controls and Motor Speed blocks.
12. Right click on red conditional terminal of While Loop to Create Constant – False Condition so the loop runs infinitely.
13. Finally run program in debug mode and control both motor's speed through increasing the knobs speed.
14. Stop program by clicking the red Abort Execution button.

NOTES:

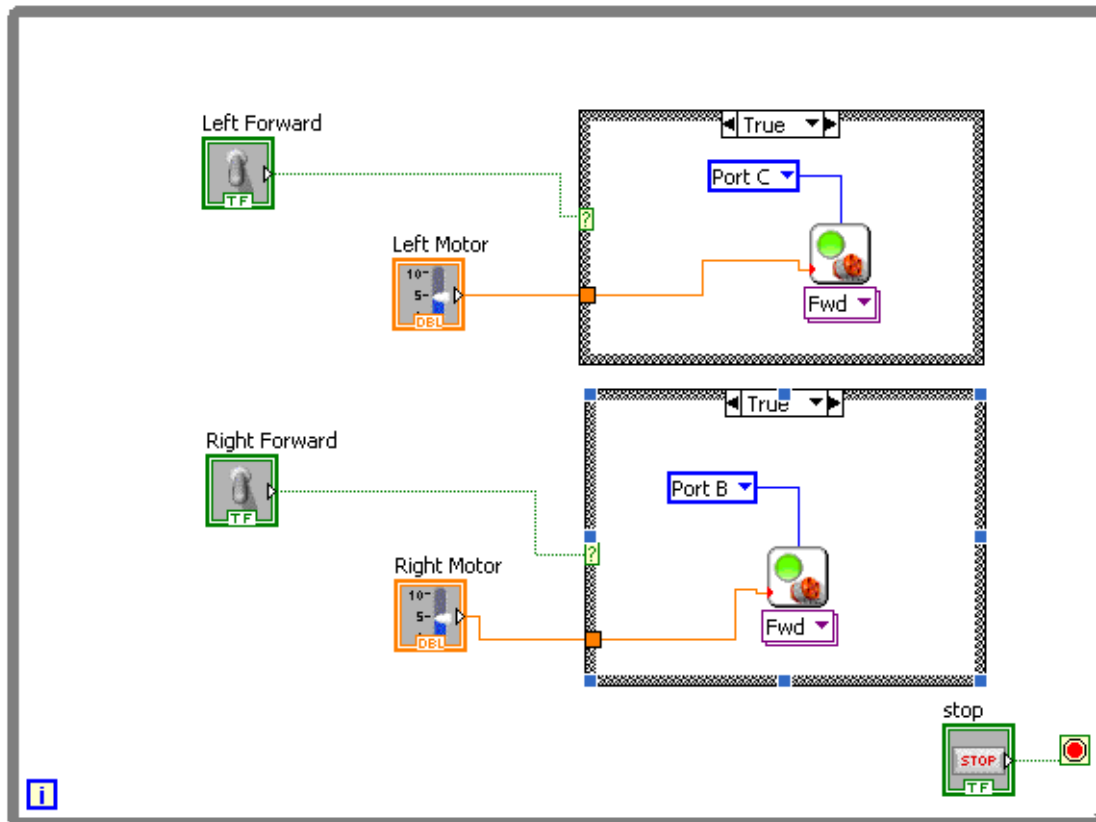


Program # 16. Advanced Front Panel Motor Control

1. Begin with a blank VI targeted to NXT.
2. While in Front Panel place two Vertical Pointer Slides from the Numeric Palette onto your workspace.
3. Enlarge the size of the knob by pulling on one of the corner dots on the knob.
4. Right click on the left Vertical Pointer Slide and select properties. Change the label from Slide to Left Motor. In the Scale Tab change the scale range maximum from 10 to 100 so the output from the slider is the proper scale to feed to the Motor later. Click OK to save properties changes.
5. Repeat the process for the other right Vertical Pointer Slide, naming it Right Motor.

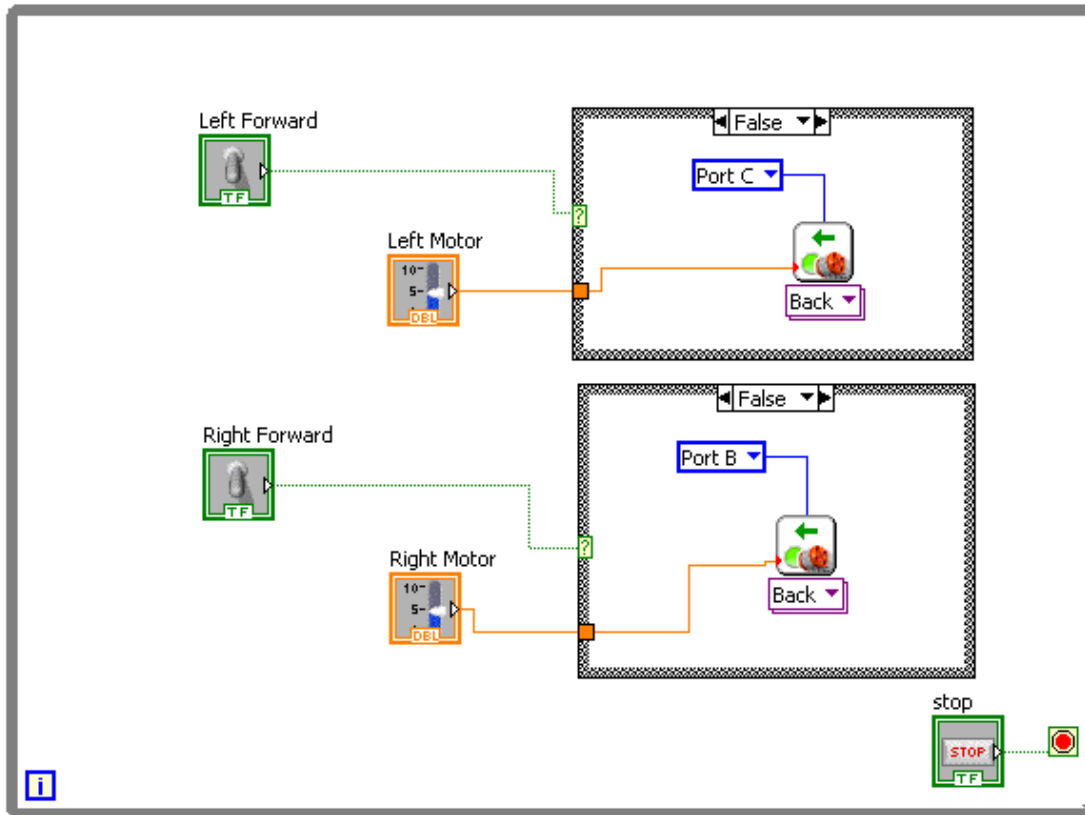


6. Navigate to the Boolean palette of the Controls palette and drag two Vertical Toggle Switches to the Front Panel. Edit the toggle switches properties to change their labels from Boolean to LEFT Forward and Right Forward. You will use these switches to control the direction each of the motors spins.
7. In the Block Diagram you should see VIs for all the controls you put on the Front Panel. Add two Motor Controls to the diagram and wire the Slide Controls to the power terminals.



8. Add ports constants to the Motor Controls and specify what port on the NXT your right and left motors are connected to.
9. Draw a Case Structure around the left Motor Control and another Case Structure around the right Motor Control.
10. Connect the Toggle Switches to the inputs of the Case Structure Selector. Now, the motors will go forward and the power given by the Sliders when the Switches are flipped up.
11. Select and copy the contents in each Case Structure and paste them in the False Case. Change the Motor Control Forward to Motor On > Reverse. Do the same for the other Case Structure.

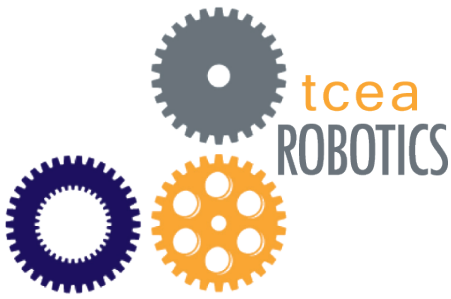
NOTES:



12. Draw a While Loop around all VIs in the Block Diagram in order to run make decisions multiple times. Right Click on the End Condition of the While Loop and click on Create Control. This puts in a Stop in the Block Diagram and a Stop Button on the Front Panel. Use this to stop the While Loop and the end the VI when it is running.

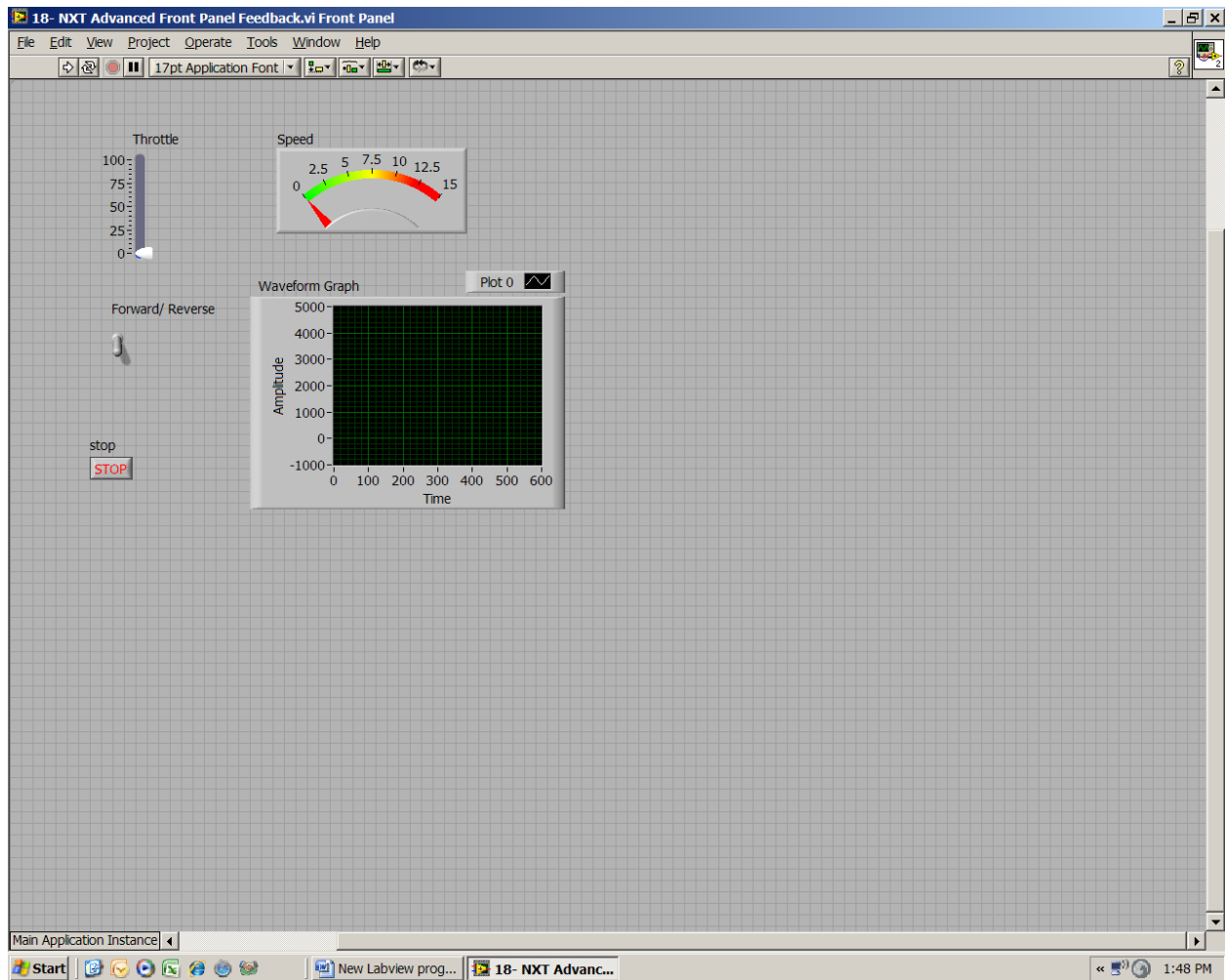
13. Return to the Front Panel and go to File>Target to Computer. Click the Run button and keep your robot plugged into the computer. You should now be able to control the robot using the controls on the Front Panel.

NOTES:



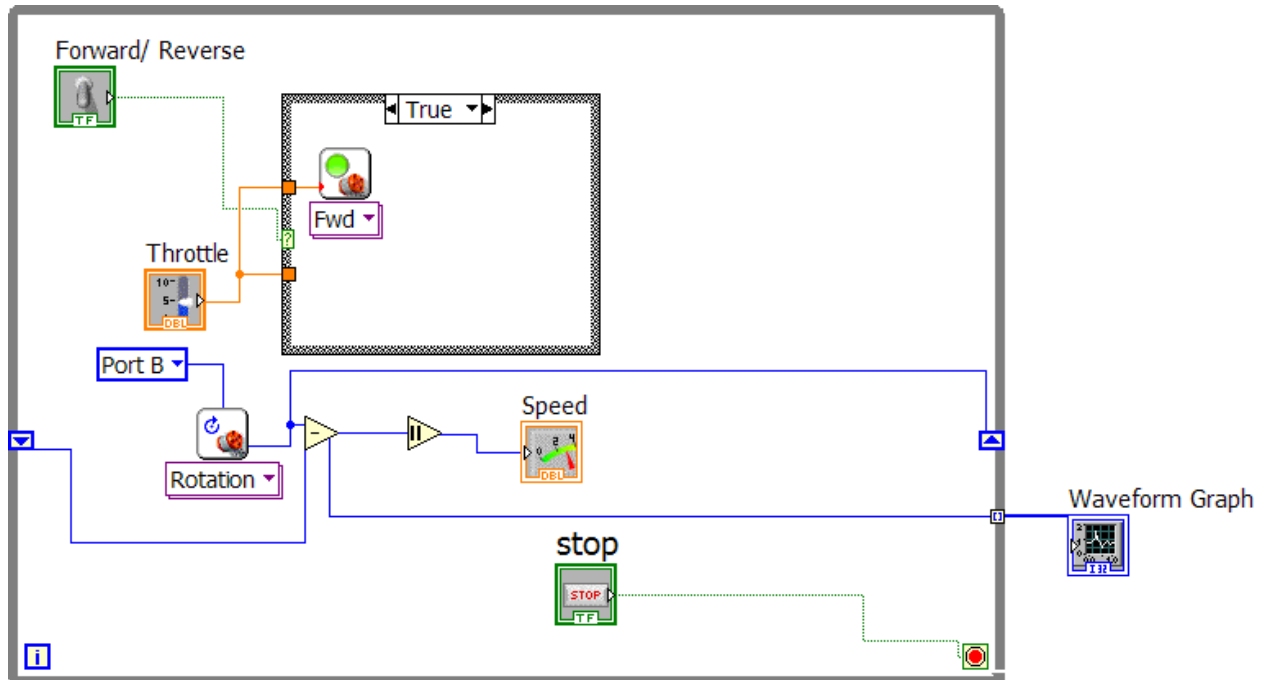
Program # 17- Advanced Front Panel Feedback

By using the Front Panel you can control the direction and throttle of the motors while looking at the resulting speed.



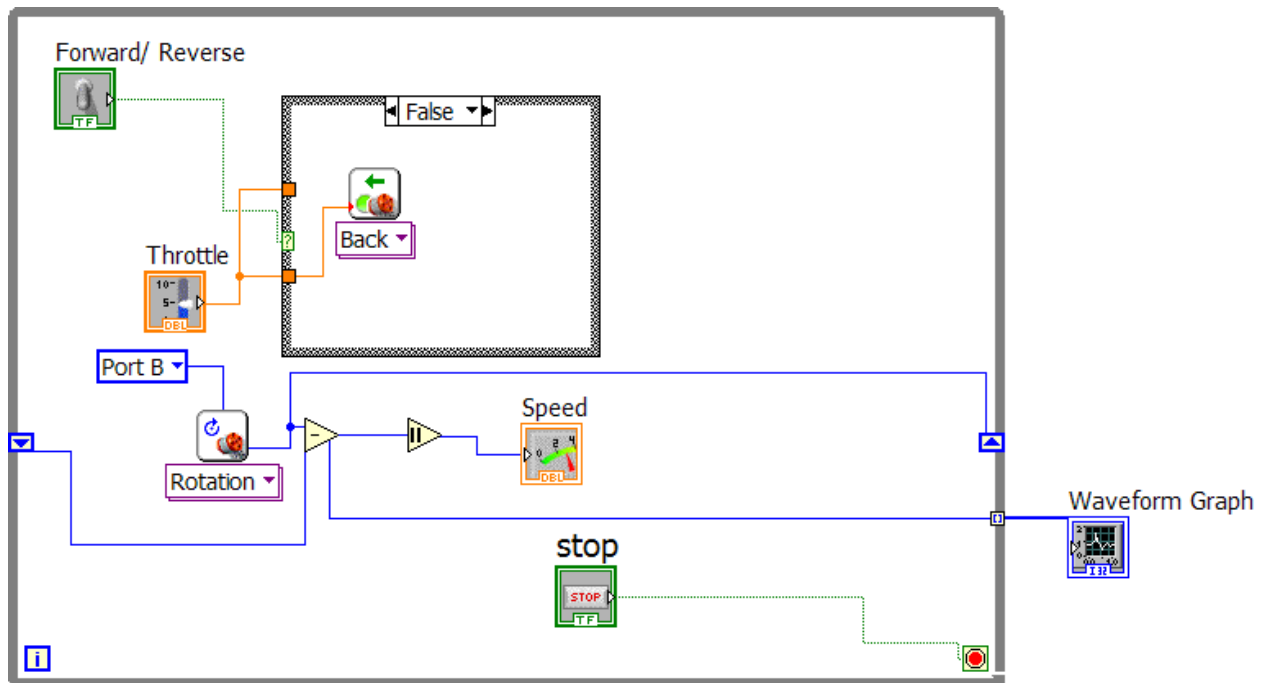
1. Begin with a blank VI targeted to an NXT. Have your robot connected via USB and turned.
2. While in Front Panel place the following controls onto the workspace:
 - a. Numeric sub-palette: Vertical Pointer Slide
Change Properties to: label Throttle and Scale Range Maximum to 100
 - b. Numeric sub-palette: Meter
Change Properties to: label Speed and Scale Range Maximum to 15

- c. Boolean sub-palette: Vertical Toggle Switch
Change Properties to: label Forward/Reverse
- d. Boolean sub-palette: Stop Button
Change Properties to: no change required
- e. Graph sub-palette: Waveform Graph
Change Properties to: no change required



4. In the Block Diagram and add a Motor Control from the NXT I/O palette. Connect the output of the Throttle to power terminal of the Motor Control. Draw a Case Structure around the Motor Control; this will be the True Case. Wire the Forward/Reverse output to the Case Structure Selector.

NOTES:



5. In the False Case place another Motor Control and use the drop down menu to change it to Motor On > Reverse and wire it like the True Case.
6. Move the Waveform Graph to the right and draw a While Loop around all the other VIs. Wire the Stop Button to the Stop Condition of the While Loop.
7. Place a Read Sensor from NXT I/O palette in the While Loop and use the drop down menu to change it to Read Rotation. To the Read Sensor add a Port Constant and set it to a port that is connected to a motor on the robot.
8. Right click on the edge of the While Loop and select Add Shift Register. This adds two Registers on the edges of the While Loop. Use the Registers to store values between loop cycles. This will find the change in rotations between cycles in order to determine speed.
9. Place a Subtract VI and an Absolute Value VI from NXT Programming>Numeric sub-palette. Wire the Rotation Sensor output to the top port of the Subtraction VI and the shift register on the right side of the While Loop. Wire the left Shift Register to the bottom port of the Subtraction VI.
10. Wire the output of the Subtraction VI to the input of the Absolute Value VI. Then, wire the output of the Absolute Value VI to the Speed Meter.
11. The program takes the previous reading of the Rotation Sensor and subtracts that from the current reading. This provides a simple estimation of the speed. Adding the Absolute Value VI shows a positive reading for the speedometer.
12. Draw a wire from the Waveform Graph VI to the wire between the Subtraction VI and the Absolute Value VI. It is correct for LabVIEW to show that the wire is broken outside of the While Loop. To rectify this broken wire, right click on the blue square where the wire is broken. Select Enable Indexing. This should eliminate the broken wire. Enable Indexing means that LabVIEW will store and keep track of the speed value at the end of each iteration of the While Loop. Once the While Loop is finished, it will give all data to Waveform Graph to display.

13. In the Front Panel select Target to Computer from the File menu then run the VI. When you are finished, press the Stop button to see a graph of your speed data.

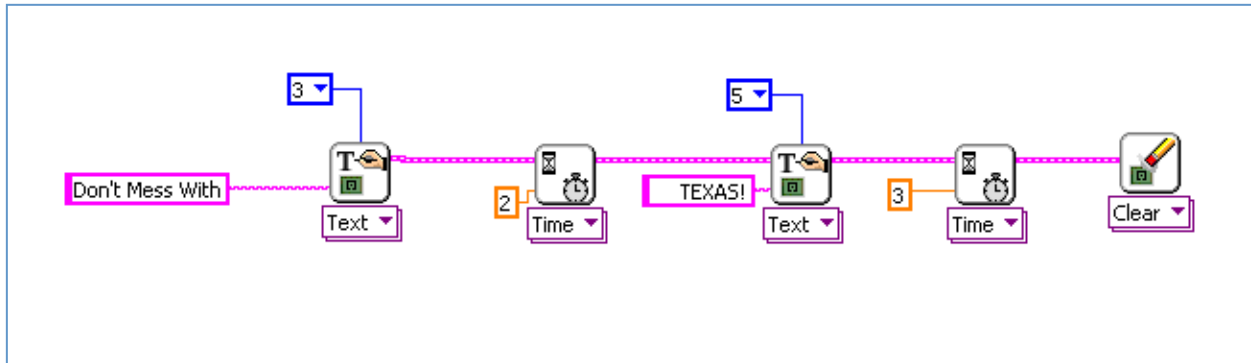
Real Life Connections that would use this program:

One example that comes to mind is a Mars Rover. Think of the Front Panel as the control center at NASA sending commands to a robot on Mars. The actual speed of the robot is communicated back to NASA so engineers can determine if the robot responded properly to the throttle commands. Perhaps this will give you an idea of the terrain (e.g., uphill or downhill)

Another is the monitoring and control of equipment in a machine shop or manufacturing plant. The engineer needs to see the speed and other parameters of the lathe that is cutting whatever part is being made in real time and must have the ability to enter various controls. All of this happens through a user interface or front panel.

NOTES:

18- NXT Basic Display – Don't Mess with TEXAS!



1. Begin with a blank VI targeted to an NXT. Have your robot connected via USB and turned.
2. This example uses 5 VIs to display “Don’t Mess with Texas!”. In this example we used two lines for the NXT display because the message would be too long to place on a single line.
3. The Display block can be found in the NXT I/O palette and use the polymorphic selector to select text.
4. This text VI displays text on the screen. The lower left input is for the text, while the top input is to specify what line to write it on. The NXT screen is divided into eight lines, so the input to the top port is a number 1 to 8.
5. The Clear VI can also be found by selecting the display block and using the polymorphic selector to select erase > screen.
6. Run the program on the NXT. The screen should display Don’t Mess with (for 2 seconds) and TEXAS! (for 3 seconds)

NOTES:



LabVIEW Reference Material Contributors:

TCEA - <http://www.tcea.org/>

LEGO Engineering - <http://www.legoengineering.com/>

National Instruments - <http://www.ni.com/>

Carnegie Mellon University - <http://www.ri.cmu.edu/>

Tufts University - <http://www.cceo.tufts.edu/>

Ben Zimmer - <http://nxtmastery.com/>

Damien Kee - <http://www.domabotics.com/>

Danny Diaz - <http://www.ni.com/>

Tony Bertucci - <http://www.lasarobotics.org/Joomla/>

Peggy Reimers - preimers@tcea.org

Betty Justus - bjustus@legoeducation.us

Jennifer Flood - jennifer@floodedu.com

Kathy Holberg - kathy@nviewcommunications.com